ACCES I/O PRODUCTS INC
10623 Roselle St., San Diego CA 92121
Tel. (858) 550-9559 FAX (858) 550-7322

EIGHT-PORT OR FOUR-PORT RS-232 SERIAL COMMUNICATION BOARD

MODELS 104-COM232-8 AND 104-COM232-4

USER MANUAL

NOTICES

The information in this document is provided for reference only. ACCES I/O Products Inc does not assume any liability arising out of the application or use of the information or products described herein. This document may contain or reference information and products protected by copyrights or patents and does not convey any license under the patent rights of ACCES, nor the rights of others.

IBM PC, PC/XT, and PC/AT are registered trademarks of the International Business Machines Corporation.

WARRANTY

Prior to shipment, ACCES equipment is thoroughly inspected and tested to applicable specifications. However, should equipment failure occur, ACCES assures its customers that prompt service and support will be available. All equipment originally manufactured by ACCES which is found to be defective will be repaired or replaced subject to the following conditions.

TERMS AND CONDITIONS

If a unit is suspected of failure, contact ACCES' Customer Service department. Be prepared to give the unit model number, serial number, and a description of the failure symptom(s). We may suggest some simple tests to confirm the failure. We will assign a Return Material Authorization (RMA) number which must appear on the outer label of the return package. All units/components should be properly packed for handling and returned with freight prepaid to ACCES' designated Service Center. After service, the units/components will be returned to the customer's/user's site freight prepaid and invoiced.

COVERAGE

First Three Years: Returned unit/part will be repaired and/or replaced at ACCES option with no charge for labor, or for parts that are not excluded by Warranty. Warranty period commences with equipment shipment.

Following Years: Throughout your equipment's lifetime, ACCES stands ready to provide on-site or in-plant service at reasonable rates similar to those of other manufacturers in the industry.

EQUIPMENT NOT MANUFACTURED BY ACCES

Equipment provided but not manufactured by ACCES is warranted and will be repaired according to the terms and conditions of the respective equipment manufacturer's warranty.

GENERAL

Under this Warranty, liability of ACCES is limited to replacing, repairing, or issuing credit (at ACCES' discretion) for any products which are proved to be defective during the warranty period. In no case is ACCES liable for consequential or special damage arising from use or misuse of our products. The customer is responsible for all changes caused by modifications or additions to ACCES equipment not approved in writing by ACCES or, if in ACCES' opinion, the equipment has been subjected to abnormal use. "Abnormal use" for purposes of this warranty is defined as any use to which the equipment is exposed other than that use specified or intended as evidenced by purchase or sales representation. Other than the above, no other warranty, expressed or implied, shall apply to any or all such equipment furnished or sold by ACCES.

TABLE OF CONTENTS

Chapter 1: INSTALLATION

INSTALLING THE SOFTWARE

You have received with your product a CD that contains all the software you need to use your board. The CD is compatible with any type of Windows or DOS system.

To install the software required for your board:

1. Insert CD in your CD ROM - If the install program does not start within 30 seconds, run “install.exe” from the root directory of the CD.
2. Click the **Install Software to Hard Disk** button.
3. Select the product you wish to install from the list shown.
4. Click **INSTALL,** or skip to #5 first.
5. The CD creates a directory with a default name; if you want to change it, click **Change** and select the path you prefer.
6. Click **Run Setup** and set the jumper options for the board.
7. Click **Finish** when finished.

You now have two types of files on your hard disk:

1. Software, including samples in C, Pascal, QuickBasic and a setup program, specifically for your board.

2. Software to help you use ACCES boards under a variety of environments:

Setup.exe	Setup program
Findbase.exe	DOS utility to determine an available base address for ISA bus , non-PnP boards. Run this program once, before the hardware is installed in the computer, to determine an available address to give the board. Once the address has been determined, run the setup program provided with the hardware to see instructions on setting the address jumpers and various option selections.
Poly.exe	A generic utility to convert a table of data into an n^{th} order polynomial. Useful for calculating linearization polynomial coefficients for thermocouples and other non-linear sensors.
Risc.bat	A batch file demonstrating the command line parameters of RISCTerm.exe.
RISCTerm.exe	A dumb-terminal type communication program designed for RS422/485 operation. Used primarily with REMOTE ACCES data acquisition Pods and our RS422/485 serial communication product line. Can be used to say

hello to an installed modem. RISCTerm stands for Really Incredibly Simple Communications TERMinal

In the *ACCES32* directory: This directory contains the Windows 95/98/NT driver used to provide access to the hardware registers when writing 32-bit Windows software. Several samples are provided in a variety of languages to demonstrate how to use this driver. The DLL provides four functions (InPortB, OutPortB, InPort, and OutPort) to access the hardware.

This directory also contains the device driver for NT. This device driver provides register-level hardware access from Windows NT, normally called through ACCES32.DLL. Two methods of using the driver are provided, the ACCES32.DLL (recommended) and the DeviceIOControl handles direct to the SYS file (slightly faster)

ACCES95 and ACCESNT

These two drivers are mentioned for users migrating from older versions of ACCES Tools. The functionality of ACCES95 and ACCESNT has been combined into ACCES32.DLL, which is described up .

In order to modify your software to use the new ACCES32.DLL, change the file you link to from ACCES95 or ACCESNT to ACCES32. No other changes are necessary.

If you want to avoid recompiling software that was written for ACCES95 or ACCESNT, just rename ACCES32.DLL to the appropriate name (95 or NT).

In the *BSAMPLES* directory: A Quickbasic sample.

In the *CSAMPLES* directory: Samples in C.

In the *PCI* directory: This directory contains PCI-bus specific programs and information. If you are not using an ACCES PCI card, you can ignore or delete this directory.

In the *PSAMPLES* directory: This directory contains samples in Pascal

In the *VBACCES* directory: Sixteen-bit DLL drivers for use with VisualBASIC 3.0 and Windows 3.1 only. These drivers provide four functions, similar to the ACCES32 DLL. However, this DLL is only compatible with 16-bit executables. Migration from 16-bit to 32-bit is simplified because of the similarity between VBACCES and ACCES32.

In the *WIN32IRQ*

directory: you have utility software to handle IRQs from any board under Win95/98 or NT

1. *Drivers.src* with 3 subdirectories

a. *DLL* Samples for using ACCES32.DLL are provided in this directory. Using this DLL not only makes the hardware programming easier (MUCH easier), but also one source file can be used for both Windows 95/98 and WindowsNT. One executable can run under both operating systems and still have full access to the hardware registers. The DLL is used exactly like any other DLL, so it is compatible with any language capable of using 32-bit DLLs. Consult the manuals provided with your language's compiler for information on using DLLs in your specific environment.

b. *SYS* The samples in this directory are provided ONLY for WindowsNT. The DeviceIOControl based interaction with the register-level driver is only available in NT. If your code is written to use this method, it will not work with Windows 95 or Windows 98.

The SYS file is the actual workhorse behind hardware access in WindowsNT. It utilizes the DeviceIOControl API function for interaction with user code. Samples are provided demonstrating this API call, but it is strongly recommended that the DLL interface be used. The DLL described above encapsulates the SYS file and performs the DeviceIOControl calls at a small penalty in speed. (A call through the DLL interface)

c. *VXD* source fo the driver

2. *Samples:* Samples in VisualC, Delphi and C++ Builder

INSTALLING THE BOARD

The 104-COM232-4 and 104-COM232-8 boards can only be installed in a PC/104 compatible slot. Before installing the board carefully read the **OPTION SELECTION** and **ADDRESS SELECTION** sections of this manual and configure the board according to your requirements. You can find an open base address with the FINDBASE program on the CD provided with the board. Finally, our setup program will lead you through the process of setting the options on the board. These must be set by jumpers on the board.

To install the board:

1. Install jumpers for selected options from either the **Option Selection** section of this manual or the suggestions of our setup software program.
2. Select the base address on the board using either the **Address Selection** section of this manual or the suggestions for 104-COM232 in our FINDBASE setup software program.
3. Turn off the computer power
4. Install the board in the PC/104 stack.
5. Install the I/O cable(s).
6. Inspect for proper fit of the board and cable and tighten mounting hardware.
9. Re-apply power and test out the board's functionality using provided samples.

Proper EMI cabling techniques (cable connect to chassis ground at the aperture, shielded twisted-pair wiring, etc) must be used for the input/output wiring.

Chapter 2: FUNCTIONAL DESCRIPTION

The 104-COM232-4/8 serial Interface board contains four/eight independent ports and provides effective RS-232 multipoint communication.

The 104-COM232-8 and 104-COM232-4 are designed in the PC/104 format.

It's dimensions are approximately 3.775 inches X 3.550 inches. All signal connections are made through two 40 pin connectors, mounted on edges of the board.

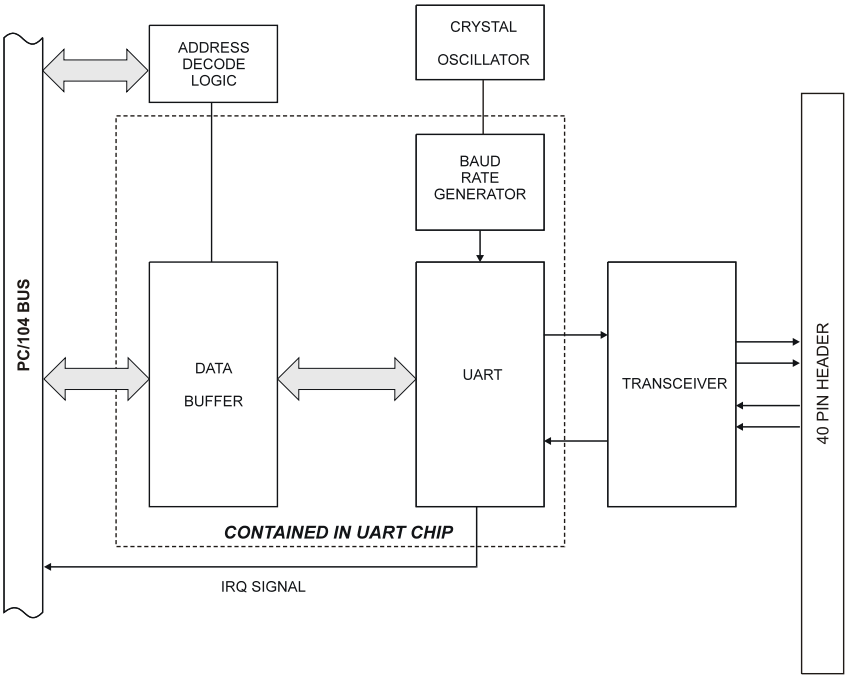
IRQ SUPPORT

The board supports the use of IRQ resources, and includes an on-board IRQ status register for use with operating systems that support this feature, such as Microsoft's Windows NT.

The status register is located at an address of the COM A address + 400H.

BLOCK DIAGRAM

(Only one serial channel shown)



Chapter 3: OPTION SELECTION

To help you locate the jumpers described in this section, refer to the OPTION SELECTION MAP at the end of this section. Operation of the serial communications section is determined by jumper installation as described in the following paragraphs.

DATA CABLE WIRING

The following connections are used to communicate between two ports (usually on different devices):

	Cable Pins		
Mode	Port 1	to	Port 2
	RX	to	TX
RS-232	TX	to	RX
	Ground	to	Ground

INTERRUPTS: The board supports interrupt levels 2, 3, 5, 7, 10, and 11 (Except those reserved by other installed hardware). The same interrupt is used for all channels, it must be entered into the interrupt location in the EEPROM.

Please note: In WindowsNT, changes must be made to the System Registry to support IRQ sharing. The following is excerpted from “Controlling Multiport Serial I/O Cards” provided by Microsoft in the MSDN library.Document id: mk:@ivt:nt40res/D15/S55FC.HTM, also available in the Windows NT Resource Kit. The text enclosed in brackets (“[]”) denotes a comment by ACCES.

The Microsoft serial driver can be used to control many *dumb* multiport serial cards. *Dumb* indicates that the control includes no on-board processor. Each port of a multiport board has a separate subkey under the CurrentControlSet\Services\Serial subkey in the Registry. In each of these subkeys, you must add values for **DosDevices**, **Interrupt**, **InterruptStatus**, **Port Address**, and **PortIndex** because these are not detected by the Hardware Recognizer. (For descriptions and ranges for these values, see Regentry.hlp, the Registry help file on the *Windows NT Workstation Resource Kit CD*.)

For example, if you have an eight-port COM board configured to use address 0x100 with an interrupt of 0x5, the values in the Registry are: [assuming every port is configured to use the same IRQ, and the addresses are configured to be consecutive and contiguous]

Serial2 subkey:
PortAddress = REG_DWORD 0x100
Interrupt = REG_DWORD 5
DosDevices = REG_SZ COM3
InterruptStatus = REG_DWORD 0x500
PortIndex = REG_DWORD 1

Serial3 subkey:
PortAddress = REG_DWORD 0x108
Interrupt = REG_DWORD 5
DosDevices = REG_SZ COM4
InterruptStatus = REG_DWORD 0x500
PortIndex = REG_DWORD 2

Serial4 subkey:
PortAddress = REG_DWORD 0x110
Interrupt = REG_DWORD 5
DosDevices = REG_SZ COM5
InterruptStatus = REG_DWORD 0x500
PortIndex = REG_DWORD 3

Serial5 subkey:
PortAddress = REG_DWORD 0x118
Interrupt = REG_DWORD 5
DosDevices = REG_SZ COM6
InterruptStatus = REG_DWORD 0x500
PortIndex = REG_DWORD 4

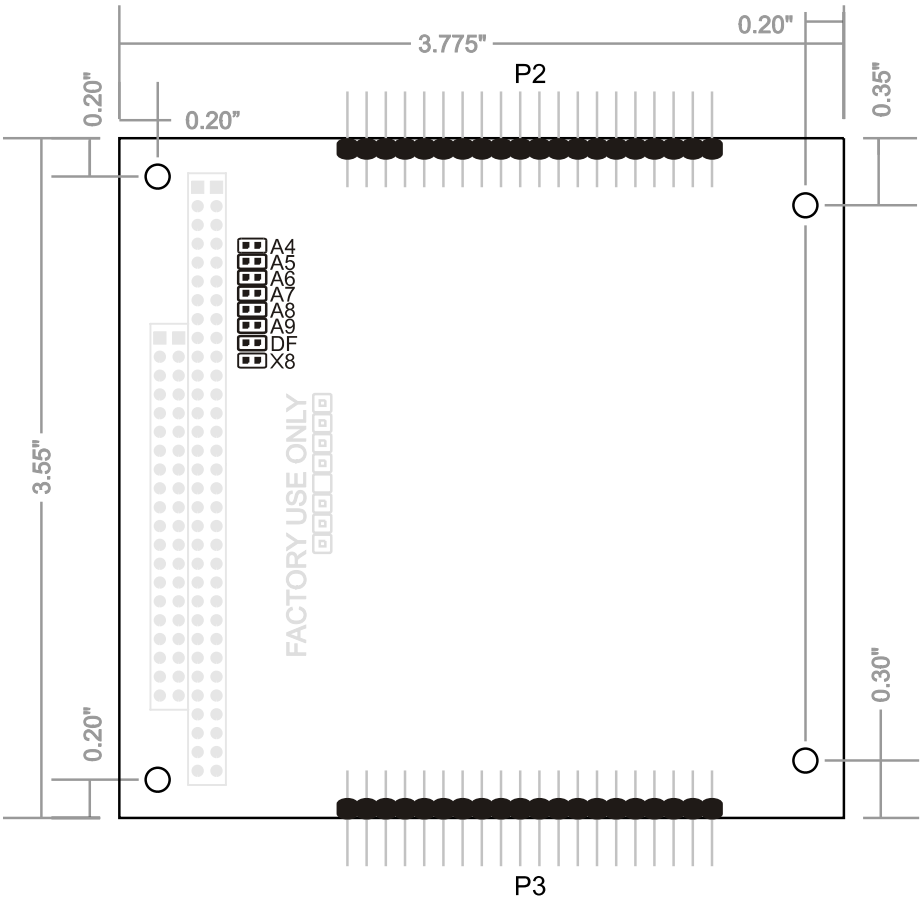
Serial6 subkey:
PortAddress = REG_DWORD 0x120
Interrupt = REG_DWORD 5
DosDevices = REG_SZ COM7
InterruptStatus = REG_DWORD 0x500
PortIndex = REG_DWORD 5

Serial7 subkey:
PortAddress = REG_DWORD 0x128
Interrupt = REG_DWORD 5
DosDevices = REG_SZ COM8
InterruptStatus = REG_DWORD 0x500
PortIndex = REG_DWORD 6

Serial8 subkey:
PortAddress = REG_DWORD 0x130
Interrupt = REG_DWORD 5
DosDevices = REG_SZ COM9
InterruptStatus = REG_DWORD 0x500
PortIndex = REG_DWORD 7

Serial9 subkey:
PortAddress = REG_DWORD 0x138
Interrupt = REG_DWORD 5
DosDevices = REG_SZ COM10
InterruptStatus = REG_DWORD 0x500
PortIndex = REG_DWORD 8

OPTION SELECTION MAP



Chapter 4: ADDRESS SELECTION

The card uses a control block base address, and each port has its own base address. Each base address can be selected anywhere within an I/O address range 100-3F8 hex, providing that the address does not overlap with other functions. If in doubt, refer to the table below for a list of standard address assignments. (The primary and secondary binary synchronous communication ports are supported by the Operating System.) The FINDBASE base address locator program provided with your card will assist you to select a base address that will avoid this conflict.

STANDARD ADDRESS ASSIGNMENTS FOR PC AND PC/XT COMPUTERS

HEX RANGE	USAGE
000-00F	DMA Chip 8237A-5
020-021	Interrupt 8259A
040-043	Timer 8253-5
060-063	PPI 8255A-5
080-083	DMA Page Register
0AX	NMI Mask Register
0CX	Reserved
0EX	Reserved
100-1FF	Not usable
200-20F	Game Control
210-217	Expansion Unit
220-24F	Reserved
278-27F	Reserved
2F0-2F7	Reserved
2F8-2FF	Asynchronous Comm'n (secondary)
300-31F	Prototype Card
320-32F	Fixed Disk
378-37F	Printer
380-38C**	SDLC Communications
380-389**	Binary Synchron's Comm'n (secondary)
3A0-3A9	Binary Synchron's Comm'n (primary)
3B0-3BF	IBM Monochrome Display/Printer
3C0-3CF	Reserved
3D0-3DF	Color/Graphics
3E0-3E7	Reserved
3F0-3F7	Diskette
3F8-3FF	Asynchronous Comm'n (primary)

** These options cannot be used together-addresses overlap

STANDARD ADDRESS ASSIGNMENTS FOR 286/386/486 COMPUTERS

Hex Range	Usage
000-01F	DMA Controller 1
020-03F	INT Controller 1, Master
040-05F	Timer
060-06F	8042 (Keyboard)
070-07F	Real Time Clock, NMI Mask
080-09F	DMA Page Register
0A0-0BF	INT Controller 2
0C0-0DF	DMA Controller 2
0F0	Clear Math Coprocessor Busy
0F1	Reset Coprocessor
0F8-0FF	Arithmetic Processor
1F0-1F8	IDE Fixed Disk
200-207	Game I/O
278-27F	Parallel Printer Port 2
2F8-2FF	Asynchronous Comm'n (Secondary)
300-31F	Prototype Card
360-36F	Reserved
378-37F	Parallel Printer Port 1
380-38F	SDLC or Binary Synchronous Comm'n 2
3A0-3AF	Binary Synchronous Comm'n 1
3B0-3BF	Monochrome Display/Printer
3C0-3CE	Local Area Network
3D0-3DF	Color/Graphic Monitor
3F0-3F7	Floppy Diskette Controller
3F8-3FF	Asynchronous Comm'n (Primary)

The address jumpers determine the address of the control block; the addresses and interrupts of the ports are taken from the onboard EEPROM. The interrupt sharing register (mainly used in NT4) is referenced to the address of Channel A.

The address bytes entered into the EEPROM represent address lines A9 thru A3. The easiest way to determine the byte to write for a desired address is to divide the address by 8. For instance, a base address of 300 would be 300/8 = 60, an address of 308/8 = 61, and so on. (All addresses are in hex.)

ADDRESS JUMPERS

	1st Digit		2nd Digit			
Jumper Label	A9	A8	A7	A6	A5	A4
Address Line Controlled	A9	A8	A7	A6	A5	A4
Hexadecimal Value	200	100	80	40	20	10

In order to read the address jumper setup, assign a binary “1" to jumpers that are not installed and a binary “0" to jumpers installed. For example, as illustrated in the following table, jumper selection corresponds to binary 10 0000 xxxx (hex 200). The “xxx” represents address lines A3, A2, A1, and A0 used on the card to select individual registers, as described in the **PROGRAMMING** section of this manual.

EXAMPLE ADDRESS SETUP

Jumper Label	A9	A8	A7	A6	A5	A4
Conversion Factors	2	1	8	4	2	1
Jumper Installed	NO	YES	YES	YES	YES	YES
Binary Representation	1	0	0	0	0	0
Hex Representation	2		0			

Review the **ADDRESS SELECTION TABLE** carefully before selecting the card address. If the addresses of two installed functions overlap you will experience unpredictable computer behavior.

Chapter 5: PROGRAMMING

The port addresses and IRQs are selected by software through a control block; the base address of the control block is selected by jumpers. The functions within the control block are shown in the control block register map below.

Control Block Register Map

Address	Read Function	Write Function
Base Address + 0	--	Enable IRQs
Base Address + 1	--	EEPROM Address
Base Address + 2	--	EEPROM Data
Base Address + 3	--	Load EEPROM To Registers

Notes:

1) Once the eeprom has been loaded, its data will be automatically transferred to the appropriate registers on power up. Therefore, it should normally only be necessary to issue the load to registers command when the data has been changed and it is desired to use the card without rebooting.

2) The IRQs are normally enabled when the first write to any UART occurs. It is not normally necessary to issue an enable IRQ command when using the card.

The addresses and IRQs of the ports are loaded from an EEPROM on the card. In addition to automatically loading them at power-on, they can be loaded by software by a write to the control block. The addresses and interrupts are stored in the EEPROM as shown on the EEPROM address map below.

EEPROM Address Map

EEPROM Address	EEPROM Data Meaning
1	Address for Channel A
2	Address for Channel B
3	Address for Channel C
4	Address for Channel D
5	Address for Channel E (-8 Only)
6	Address for Channel F (-8 Only)
7	Address for Channel G (-8 Only)
8	Address for Channel H (-8 Only)
9	IRQ for Channel A

A	IRQ for Channel B
B	IRQ for Channel C
C	IRQ for Channel D
D	IRQ for Channels E,F,G,H (-8 Only)

As mentioned elsewhere, the addresses entered represent A3 - A9. Therefore, the data entered is the desired address, divided by 8.

The IRQ entry is the number of the desired IRQ.

When the card is first installed in a system, the ports are not necessarily at unused addresses. To prevent conflicts with other devices in the system, the card has a jumper that disables the ports, next to the base address jumpers and labeled “DF”. The control block remains enabled in this mode, allowing software to set the port addresses appropriately. When the DF jumper is then removed, the ports will then be at the configured addresses.

To write data to the EEPROM, first write the address to the EEPROM Address register, then write to or read from the EEPROM Data register. The data transfer inside the card takes 2 milliseconds, during which the card should not be accessed. For example, to set Channel A to address 3F8, IRQ 5, with the control block base address set to 200 (by jumpers):

Write 01 to 201.
Write 7F to 202.
Wait 2ms.
Write 09 to 201.
Write 05 to 202.
Wait 2ms.
Then write anything to 203 to start using these values.

All data may be entered into the EEPROM and then written to the appropriate registers with a single write to base address + 3.

SAMPLE PROGRAMS

Sample programs are installed with the CD that ships with the 104-COM232-4/8 card. See the CD for details.

INITIALIZATION

Initializing the chip requires knowledge of the UART's register set. The first step is to set the baud rate divisor. You do this by first setting the DLAB (Divisor Latch Access Bit) high. This bit is Bit 7 at Base Address +3. In C code, the call would be:

```
outportb(BASEADDR +3,0x80);
```

You then load the divisor into Base Address +0 (low byte) and Base Address +1 (high byte). The following equation defines the relationship between baud rate and divisor:

desired baud rate = (clock frequency) / (16 * divisor)

Clock frequencies of 1.8432 MHz (Standard) and 14.7456 MHz (X8) are provided. Below is a table for the popular divisor frequencies:

BAUD RATE DIVISOR VALUES

Baud Rate	Divisor (Std)	Divisor (X8)	Notes
921600	-	1	
460800	-	2	
230400	-	4	
115200	1	8	
57600	2	16	
38400	3	24	
28800	4	32	
19200	6	48	
14400	8	64	
9600	12	96	Most Common
4800	24	192	
2400	48	384	
1200	96	768	

* RS-232 communication lines have a maximum length of 50 feet, regardless of speed.

In C, the code to set the chip to 9600 baud is:

```
outportb(BASEADDR, 0x0C);
outportb(BASEADDR +1,0);
```

The second initializing step is to set the Line Control Register at Base Address +3. This register defines word length, stop bits, parity, and the DLAB.

Bits 0 and 1 control word length and allow word lengths from 5 to 8 bits. Bit settings are extracted by subtracting 5 from the desired word length.

Bit 2 determines the number of stop bits. There can be either one or two stop bits. If Bit 2 is set to 0, there will be one stop bit. If Bit 2 is set to 1, there will be two stop bits.

Bits 3 through 6 control parity and break enable. They are not commonly used for communications and should be set to zeroes.

Bit 7 is the DLAB discussed earlier. It must be set to zero after the divisor is loaded or else there will be no communications.

The C command to set the UART for an 8-bit word, no parity, and one stop bit is:

```
outportb(BASEADDR +3, 0x03)
```

The final initialization step is to flush the receiver buffers. You do this with two reads from the receiver buffer at Base Address +0. When done, the UART is ready to use.

RECEPTION

Reception can be handled in two ways: polling and interrupt-driven. When polling, reception is accomplished by constantly reading the Line Status Register at Base Address +5. Bit 0 of this register is set high whenever data are ready to be read from the chip. A simple polling loop must continuously check this bit and read in data as it becomes available. The following code fragment implements a polling loop and uses a value of 13, (ASCII Carriage Return) as an end-of-transmission marker:

```
do
{
        while (!(inportb(BASEADDR +5) & 1)); /*Wait until data ready*/
        data[i++]= inportb(BASEADDR);
}
while (data[i]!=13);  /*Reads the line until null character rec'd*/
```

Interrupt-driven communications should be used whenever possible and is required for high data rates. Writing an interrupt-driven receiver is not much more complex than writing a polled receiver but care should be taken when installing or removing your interrupt handler to avoid writing the wrong interrupt, disabling the wrong interrupt, or turning interrupts off for too long a period.

The handler would first read the Interrupt Identification Register at Base Address +2. If the interrupt is for Received Data Available, the handler then reads the data. If no interrupt is pending, control exits the routine. A sample handler, written in C, is as follows:

```
readback = inportb(BASEADDR +2);
if (readback & 4)        /*Readback will be set to 4 if data are available*/
                data[i++]=inportb(BASEADDR);
outportb(0x20,0x20);        /*Write EOI to 8259 Interrupt Controller*/
return;
```

TRANSMISSION

To transmit a string of data, the transmitter must first check Bit 5 of the Line Status Register at Base Address +5. That bit is the transmitter-holding-register-empty flag. If it is high, the transmitter has sent the data. The process of checking the bit until it goes high followed by a write is repeated until no data remains.

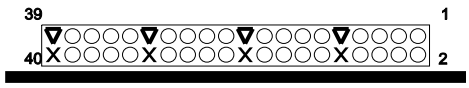
The following C code fragment demonstrates this process:

```
outportb(BASEADDR +4, inportb(BASEADDR +4)|0x02);
                        /*Set RTS bit without altering states of other bits*/
while(data[i]);                 /*While there is data to send*/
        {
        while(!(inportb(BASEADDR +5)&0x20));    /*Wait until transmitter is empty*/
        outportb(BASEADDR,data[i]);
        i++;
        }
outportb(BASEADDR +4, inportb(BASEADDR +4)&0xFD);
                        /*Reset RTS bit without altering states of other bits*/
```

Chapter 6: CONNECTOR PIN ASSIGNMENTS

Two popular 40 pin IDC male connectors are used for interfacing to communication lines.

The pins are numbered on each connector as shown:



Pin Type Key
- ▽ Ground
- x Not Connected
- ○ Serial Signal

Connector pin assignments are as follows:

Connector P2 (4- and 8-port boards)

Pin	Ch	Function	Pin	Ch	Function
1	A	CD	21	C	CD
2	A	DSR	22	C	DSR
3	A	RX	23	C	RX
4	A	RTS	24	C	RTS
5	A	TX	25	C	TX
6	A	CTS	26	C	CTS
7	A	DTR	27	C	DTR
8	A	RI	28	C	RI
9	A	Ground	29	C	Ground
10			30		
11	B	CD	31	D	CD
12	B	DSR	32	D	DSR
13	B	RX	33	D	RX
14	B	RTS	34	D	RTS
15	B	TX	35	D	TX
16	B	CTS	36	D	CTS
17	B	DTR	37	D	DTR
18	B	RI	38	D	RI
19	B	Ground	39	D	Ground
20			40		

Connector P3 (8-port board only)

Pin	Ch	Function
1	E	CD
2	E	DSR
3	E	RX
4	E	RTS
5	E	TX
6	E	CTS
7	E	DTR
8	E	RI
9	E	Ground
10		
11	F	CD
12	F	DSR
13	F	RX
14	F	RTS
15	F	TX
16	F	CTS
17	F	DTR
18	F	RI
19	F	Ground
20		

Pin	Ch	Function
21	G	CD
22	G	DSR
23	G	RX
24	G	RTS
25	G	TX
26	G	CTS
27	G	DTR
28	G	RI
29	G	Ground
30		
31	H	CD
32	H	HSR
33	H	RX
34	H	RTS
35	H	TX
36	H	CTS
37	H	HTR
38	H	RI
39	H	Ground
40		

Chapter 7: SPECIFICATION

COMMUNICATIONS INTERFACE

Two 40-pin connectors are provided on the 8-port board.
One 40-pin connector is provided on the 4-port board.

ENVIRONMENTAL

Operating Temperature Range: 0 to +60 °C
Storage Temperature Range: -50 to +120 °C
Humidity: 5% to 95%, non-condensing.

Power Required: +5 VDC at 400 mA typical, 800 mA maximum.

Size: PC/104 format, 3.5” by 3.75”.